QACA: Quality Assured Context Acquisition in Context-Aware Computing

Yuanping Li, Ling Feng, Tharam Dillon

Department of CS&T, Tsinghua University, Beijing, China

Digital Ecosystem and Business Intelligence Institute, Curtin University of Technology, Australia

{liyp04@mails., fengling@tsinghua.edu.cn, Tharam.Dillon@cbs.curtin.edu.au


Abstract

Previous research work in context quality primarily focuses on identifying and modeling it. However, assuring context quality is no less challenging than modeling it. Difficulties lie in acquiring high-quality context from the context providers whose quality and availability cannot be guaranteed. In this paper, we propose a context quality assurance strategy QACA based on the least square error, by leveraging the redundancy of context providers. In this strategy, redundant imprecise and unstable context providers are used to constitute a more precise and stable context provision. Such a strategy provides a theoretical basis to implement a specification-based context acquisition middleware.

Keywords: Context Acquisition, Context Quality, Uncertain Context

1. Introduction

To help people access information more effectively, “context” is introduced in ubiquitous computing. Context is “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [1]”. Interesting context-aware systems appeared in tour guide, smart place, health monitor, content adaptation on mobile devices, office and conference assistant, communication assistant, etc. For example, in the museum of Cooltown [2], a visitor holds a PDA which can display the exhibits’ information when the visitor approaches the exhibits. In MusixMix [3], when a person enters the room, his/her preferred music can be played automatically. In FireGuide [6], the context-aware guide from the speakers and mobile devices can help people leave the dangerous place.

Timely and high-quality context is critical to the success of these context-aware applications [7]. In fact, almost all the context-aware systems have two indispensable components - context provider and context consumer. Context acquisition from the provider to the upper consumer relates to the lowest layer of the context-aware application programming model [33] and is included in the fundamental tiers in the context-aware system architecture [8].

In context acquisition, one of the main challenges is how to manage the data flow from context providers to consumers. The problem is non-trivial when 1) the context providers and consumers are heterogeneous and distributed; 2) the quantities of context providers and consumers are large. The method of unifying context acquisition interface was adopted in some previous context aware frameworks. However, such a method has the difficulties in reusability and compatibility among different frameworks. To address these difficulties, a specification-based middleware solution was proposed in [7]. System developers only have to configure and update the specification, and the acquisition requirement is fulfilled by the middleware automatically. An important functionality of such a specification-based method is to enable system developers to set the context quality requirement in the specification, such as context precision. However, we note that how to assure the quality requirement of context is not well addressed in the previous researches.

This paper proposes a quality assurance strategy wherein the main idea is to use redundant imprecise and unstable context providers to constitute a more precise and stable context provision. Furthermore, to facilitate the modeling of context acquisition requirement and process, we propose a graphical design model for the system developers. Hence, the contributions of this paper are twofold.
1) A context acquisition model is introduced to enhance the representation of context acquisition specification.
2) The paper identifies some mechanisms to assuring the context quality, e.g., a least square error method for context precision and a verification method for redundancy degree.

The remainder of the paper is organized as follows. Section 2 describes the related work in context acquisition middleware. Section 3 describes the usage of context acquisition diagram to model the context acquisition requirement. Section 4 introduces the quality assurance strategy by minimizing the error square. Section 5 analyzes the influence of context provider failure and context redundancy degree. Section 6 concludes the paper.

2. Related work

2.1. Challenges in context acquisition

As pointed by Zhdanova et al. [9], there are two salient challenges in context acquisition. The first challenge is to acquire high-level context from low-level hardware or software sensors, e.g., to track the location or to estimate the activity of a person by using load sensors [10]. The second challenge is to manage the distribution, heterogeneity, and scalability issues in context acquisition. It is not rare that the context providers are distributed and heterogeneous in a context aware system. In effect, context consumers can also be distributed and heterogeneous.

2.2. Context acquisition middleware

Researchers have proposed many frameworks or middlewares for context aware computing. Many of them eventually did not focus on context acquisition, but as a whole framework for building context-aware applications. However, they usually incorporate context acquisition as a component.

One of the main ideas of these researches is to unifying the context acquisition interfaces in the framework. Dey et al. defined acquisition interfaces for context widgets that mediate between the appliance and the environment [11]. One such interface is shown in Figure 1.

```java
public Error subscribeToHandler(String subid, String remoteHost, int remotePort, String remoteld, String callback, String url, Conditions conditions, Attributes attributes)
```

Figure 1. A context acquisition interface in context widget

Dey et al.’s work is a pioneering study on context-aware architecture. Hong built an infrastructure for context-aware computing called ContextFabric, which focused on context data modeling, context specification language, and protection mechanism for safeguarding privacy needs [12]. On top of the ContextFabric, a query processing service called liquid for distributed continuous query processing of context data [13]. Chen et al. built a context-aware platform called Solar. In the platform, contextual data sources are in the form of stream publishers, and a peer-to-peer overlay is used to support data-driven services [5]. Martinez and Salavert also defined some APIs to set frequency and cache size in the API [31]. Presecan and Tomai provided a middleware architecture in order to query context information in a standard REST style queries [14]. Yamabe et al. developed a framework called Citron to acquire context data for multiple sensory personal devices [34]. The work focused on the mobile phone side rather than on the server side.

Another trend of the context acquisition research is to make the context providers as services. Advantages exist in a service-oriented platform, which provides the functionalities of service registration, discovery, and composition. Yang et al. implemented the context acquisition function as a
kind of Web service (GPS, RFID) [15]. Gui et al. used a server as a repository of context providers. Context providers are registered and managed by the server [16]. Costa et al. introduced a service architecture to support context-aware applications [17], in which a subscription language was defined to configure the platform to react to a given correlation of events, potentially involving contextual information. Gu et al. also proposed a service-oriented middleware for building context-aware service [18]. Costa and Botelho generalized Dey et al.’s widget method to a service-oriented platform [19]. Ritchie proposed the peripheralware that wrapped the context service functionalities to tackle the issues of access control, cost/importance trade offs, visibility, handing off requests, and minimizing user interruption [20]. Recently, He et al. illustrated that the context information could also be provided as a service in a cloud infrastructure [21].

If the framework only unifies the context acquisition interfaces, the developers still need to take efforts to implement the interfaces for different context providers, e.g., hardware or software sensors. The incompatibility of different frameworks also makes context information integration difficult. Recent research [7] proposed specification based method to help system developers to fulfill the acquisition functionality by directly using standard provision protocols and consumption models. Some techniques can be utilized in the specification based method to automatically select context providers for the context consumers, e.g., the similarity based mapping proposed by Xue et al. [22]. To implement an efficient context acquisition middleware, some acquisition performance optimization techniques were discussed for Web service based context providers and ontology based context consumers in [23]. The conditions for data correctness were also discussed.

2.3. Quality of Context

Sheikh et al. identified several indicators of quality of context. They include precision, freshness, spatial resolution, temporal resolution [24, 25]. Broens et al. described the techniques of dynamic context binding, by storing context types, frequency, context values in the database. A user inputted the context quality requirement and the system searched the database for the context provider to bind [26].

However, we find that 1) it has not been described in the literature yet how to model the context acquisition with graph representations from the data flow perspective; 2) controlling and improving the quality of context in a specification based context acquisition middleware is still not well addressed, especially when any single context provider could not meet the quality requirement, whilst the importance of modeling quality of context has been mentioned in the literature.

3. Model and middleware for context acquisition

In a context-aware application development platform, context acquisition component manages collecting context data from the context providers to the context consumers. The position of acquisition part is in the middle of context providers and context consumers, as illustrated in Figure 2. Note that the relationships among context reasoning, context history management and other components are not the focus of this paper, so they are not explicitly shown in the figure. The physical locations of context providers, context consumers, and context acquisition middleware can be distributed.

![Figure 2. The position of context acquisition in the context-aware system architecture](image-url)
Li and Feng proposed a text language for context acquisition in [7]. We briefly review some definitions that will be used in this paper.

**Definition 3.1** *Context space* of a context with several components is a set of all possible values of the context.

**Definition 3.2**: A *context provider* $p$ is a tuple $(P_{name}, P_{protocol}, P_{method})$, where $P_{name}$ is the name of $p$, $P_{protocol}$ indicates the communication protocol of $p$ (such as USB, R232 serial port, Bluetooth, or RMI used to provide context data, etc.), and $P_{method}$ is the method used to acquire context value from $P$ via the communication protocol $P_{protocol}$. Different communication protocols correspond to different methods. For instance, when $P_{protocol}=$USB, $P_{method}$ details the device information, data packet format, etc. When $P_{protocol}=$RMI, $P_{method}$ details the registry URL of the provider server, method name, and an argument list.

**Definition 3.3**: A *context consumer* $c$ is a tuple $(C_{name}, C_{type}, C_{method})$, consisting of the name $C_{name}$, the type $C_{type}$ (e.g., ontology model, relational model, etc.) $C_{method}$ defines the method by which the context consumer handles the received context data. In an ontology-based context consumer, $C_{method}$ details the data location and operation. For example, the data location is Room601’s property hasTemperature and the operation is to update the data.

**Definition 3.4** If changes of context data from context provider $p$ might cause changes of the context received by context consumer $c$, $p$ is *connected with* $c$.

To facilitate the designing of the context acquisition functionality, we further propose to add some graphical modeling ability herein, which is called context acquisition diagram. The system developers can deal with the graphical model rather than only a text specification. The graphical model can be stored in a file, or in a database. The model can be updated in the runtime, which dynamically changes the functionality of context acquisition. A usage scenario of context acquisition middleware is illustrated in Figure 3.

![Figure 3](image-url)  
**Figure 3.** The usage scenario of context acquisition middleware

In the context acquisition diagram, the logical locations and relationships of context providers and consumers are drawn. The model can be transformed to a text specification. There are four basic graphical elements in the context acquisition, which denote the context provider, context consumer, connector, and dependency, as shown in figure 4. The arrow of the connector indicates the direction of context data flow. The arrowed lines are the connectors, which show some relationships existing between the connected context provider and consumer. To illustrate the detailed relationships, a triangle which stands for context dependency is used. Designers can add descriptions to triangles, which give the detailed account of some acquisition constraints and quality requirements.

![Figure 4](image-url)  
**Figure 4.** The basic graphical elements of the context acquisition model
Figure 5 shows some examples of context acquisition diagrams. All dependency triangles are one-to-one connected to context consumers. \( DS_i \) denotes a set of context providers. They are connected to the \( i \)th dependency triangle. Through the triangle, they further connect to a context consumer. For example, in Figure 5a, two context providers (GPS and WiFi) are connected to one location consumer. Let \( DS = [DS_1, ..., DS_n]^T \). The topology of a context acquisition can be expressed as

\[
(DS_{\text{acq}}) = MP_{\text{nxl}}
\]

Where \( M \) is \( n \times m \) matrix with field \{0,1\} and \( P \) is the vector of context providers. And the addition operations involved in the matrix multiplication are set additions. For Figure 5a,

\[
DS_{\text{acq}} = (1 \ 1)P_{\text{2x1}}
\]

Another example is Figure 5b. The consumer, which uses RDF to modeling context, receives the context data from the channels of a data acquisition card.

\[
DS_{\text{acq}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}P_{\text{3x1}}
\]

Figure 5c shows a bearing degree received by the context consumer, which is computed by adding the base and offset from two providers. Figure 5d expresses the location coordinates of mobile phones, which are sent to a stream processing engine with a relational schema (id, x, y, timestamp).
The specification of the whole system can be distributed in different computing devices or centered in one device. The context providers, context consumers and the specifications all may be shared by different acquisition middleware instances. The context providers, context consumers can be organized in a pere-to-pere architecture.

3.3. Quality requirement

System developers may need to specify quality requirement for the acquisition middleware. As shown in Figure 5b, the system developer needs the acquired temperature to be less than 1 degree deviating from the real temperature in a room, in 95.44% cases. However, sometimes context providers are subject to noise disturbance, and the sensors are likely to fail to work. In the example, the best temperature sensor in the system may only provide a temperature that is less than 2 degrees deviating from the real temperature in 95.44% cases.

4. Quality assurance strategy for uncertain context

Uncertainty of context incurs difficulties of assuring quality of context. In this section, we discuss a unified quality assurance formulation by minimizing the square error. Then, the focus is on the analysis of context data with continuous values.

4.1. Unified least square error based formulation

To assure high context quality, we introduce the redundancy of context providers. Let \( c \) denote a context consumer. The real context value that the context consumer \( c \) needs is \( y \). There is a set of context providers \( \{ p_1, \ldots, p_m \} \). Each \( p_i, i = 1, \ldots, m \), can provide context value \( x_i \) for the context consumer \( c \). Assume the statistical expected value of \( x_i \) is \( y \) which is unknown. Then, we can estimate \( y \) with least square error. That is to find the solution of \( y \) through

\[
\min \sum_{i=1}^{m} d(y, x_i)^2
\]

where \( d(y, x_i) \) is the distance between \( y \) and \( x_i \).

4.2. Quality assurance for context data

Suppose \( x_i \) is from a context space with continuous values. The variance of \( x_i \) is \( \sigma_i^2 \). Then, the least square error formulation becomes

\[
\min \sum_{i=1}^{m} (y - x_i)^2
\]

Let \( f = \sum_{i=1}^{m} (y - x_i)^2 \). Since \( f \) is convex, by solving the first order condition \( \frac{\partial f}{\partial y} = 0 \), we have the minimum of \( f \) when \( y = \frac{\sum_{i=1}^{m} x_i}{m} \).

We introduce the variance here as an indicator of context uncertainty in the quality requirement. The system developers can set a required precision bound with a confidential probability, as in the Figure 5b. Alternatively, they can also set the required variance directly. In this paper, we suppose the context
data comply with some distributions. For instance, suppose the temperature measurement complies with a normal distribution, we have

\[ \frac{y - \mu}{\sigma} \sim N(0,1) \]

For a normal distribution, we know that \( \Pr\{y \in [\mu - 2\sigma, \mu + 2\sigma]\} = 95.44\% \). Then, the precision bound can be transformed to a variance requirement. In the above example, we have \( 2\sigma = 2 \), i.e., \( \sigma = 1 \).

If we assume \( x_1, \ldots, x_m \) are independent, the variance \( \sigma^2_y \) of \( y = \frac{\sum_{i=1}^m x_i}{m} \) is \( \frac{\sum_{i=1}^m \sigma^2_i}{m} \).

**Example 4.1** For the special case where \( \sigma^2_i = \sigma^2 \), we have \( \sigma^2_y = \frac{\sigma^2}{m} \).

The above analysis shows that by incorporating \( m - 1 \) redundant context providers, the variance of \( y \) decrease to \( \frac{1}{m} \) of that of one context provider. Theoretically, any small variance requirement can be satisfied by increasing \( m \). Furthermore, in the case where context providers have different precisions, a weighted least square error can be used. That is

\[
\min \sum_{i=1}^m a_i (y - x_i)^2
\]

Similarly, we have \( y = \frac{\sum_{i=1}^m a_i x_i}{\sum_{i=1}^m a_i} \). The variance of \( y \) is \( \sigma^2_y = \frac{\sum_{i=1}^m a_i^2 \sigma^2_i}{(\sum_{i=1}^m a_i)^2} \). If \( \sum_{i=1}^m a_i = 1 \), then

\[
\sigma_y^2 = \sum_{i=1}^m a_i^2 \sigma_i^2
\]

**Example 4.2** There are two context providers \( p_1 \) and \( p_2 \) which provide context values \( x_1 \) and \( x_2 \). The variance of \( x_1 \) is 1 and the variance of \( x_2 \) is 4. \( a_1 = \frac{2}{3} \) and \( a_2 = \frac{1}{3} \). Then

\[
y = \frac{2}{3}x_1 + \frac{1}{3}x_2\]

\[
\sigma^2_y = \frac{4}{9}\sigma^2_1 + \frac{1}{9}\sigma^2_2 + \frac{4}{9} + \frac{4}{9} = \frac{8}{9}
\]

In this example, we can see that the variance of \( y \) is less than both the variances of \( x_1 \) and \( x_2 \).

In [27], Hackett and Shah introduced the Bayesian approach, which calculate the best estimate of object property in data fusion. Using similar type of calculation, we can minimize the variance of \( y \) and obtain the optimal weights \( a_1, \ldots, a_m \). The first order optimization condition is

\[
\frac{\partial g}{\partial a} = 0,
\]

where \( g = \sum_{i=1}^m a_i^2 \sigma^2_i \), \( \bar{a} = [a_1, \ldots, a_m] \), \( \sum_{i=1}^m a_i = 1 \).

We have
QACA: Quality Assured Context Acquisition in Context-Aware Computing
Yuanping Li, Ling Feng, Tharam Dillon
Journal of Convergence Information Technology, Volume 6, Number 1. January 2011

\[
\begin{align*}
\left( \sigma_1^2 + \sigma_m^2 \right) a_1 + \sigma_m^2 a_2 + ... + \sigma_m^2 a_{m-1} &= \sigma_m^2 \\
\sigma_m^2 a_1 + \left( \sigma_2^2 + \sigma_m^2 \right) a_2 + ... + \sigma_m^2 a_{m-1} &= \sigma_m^2 \\
& \vdots \\
\sigma_m^2 a_1 + \sigma_m^2 a_2 + ... + \left( \sigma_{m-1}^2 + \sigma_m^2 \right) a_{m-1} &= \sigma_m^2 \\
a_1 + a_2 + ... + a_{m-1} + a_m &= 1
\end{align*}
\]

let \( \bar{E} = \begin{pmatrix} \sigma_1^2 & \sigma_2^2 & ... & \sigma_m^2 \\ \sigma_2^2 & \sigma_2^2 & ... & \sigma_m^2 \\ \sigma_m^2 & \sigma_m^2 & ... & \sigma_m^2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \), then \( \bar{E} \bar{a} = \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_m^2 \\ 1 \end{pmatrix} \). If \( \bar{E} \) is nonsingular, we have

\[
\bar{a} = \bar{E}^{-1} \begin{pmatrix} \sigma_1^2 \\ \sigma_2^2 \\ \sigma_m^2 \\ 1 \end{pmatrix}
\]

When the variance of \( y \) is specified, the main task is to compute the configuration of context providers. For instance, there are \( k \) kinds of context providers. Each kind of context provider has the variance \( \sigma_i^2 \). There is a bound \( n_j = [n_{j1}, ..., n_{js}] \) for the number of each kind of context provider. The cost for purchasing or maintaining of one of the \( i \) th kind context provider is \( b_i \), and we have a total cost bound for purchasing or maintaining context providers, which is \( C_b \). Let the number of the \( i \) th kind of context provider be \( n_i \). Then, the quality requirement is transformed to

\[
\min \sum_{i=1}^{k} n_i a_i^2 \sigma_i^2 \\
\text{s.t.} \quad n_i \leq n_{ji}, i = 1, ..., k \\
\sum_{j=1}^{k} n_i a_i = 1 \\
\sum_{j=1}^{k} n_i b_i \leq C_b
\]

It can be solved by integer solvers, e.g. the branching and bound method [28]. The context value that the context consumer use is

\[
y = \frac{\sum_{j=1}^{k} \left( a_i \sum_{j=1}^{n} x_j \right)}{\sum_{j=1}^{n} n_i a_i}
\]
Example 4.3  Consider the special case when all context providers have the same variance $\sigma_x^2$. Meanwhile, there is no bound for the number of context providers. The specified variance for $y$ is $\sigma_y^2$. Then $\sigma_y^2 = \frac{\sigma_x^2}{m} \leq \sigma_x^2$. To meet the specification, we have $m \geq \frac{\sigma_x^2}{\sigma_y^2}$. Figure 6(a) shows the least required number of context providers, with different $\sigma_x^2$ and $\sigma_y^2$.

Example 4.4  There are two kinds of context providers. The bounds for the numbers of providers are $n_{s_1} = 10$, $n_{s_2} = 10$. The cost for one context provider $b_1 = 15$, $b_2 = 5$. The variance of the two kinds of context providers is $\sigma_1^2 = 1$, $\sigma_2^2 = 2$. We can obtain the minimum of $\sigma_y^2 = 0.0769$ at $n_1 = 3$, $n_2 = 10$, $a_1 = a_2 = 0.0769$. In this example, figure 6(b) shows the minimum of $\sigma_y^2$ when the bounds for the number of providers vary, and figure 6(c) shows the minimum of $\sigma_y^2$ when the costs for context providers vary.

5. Analysis of other factors in quality assurance
5.1. Failure of context providers

In this section, we introduce the handling of failure of context provider and context consumer.

**Definition 5.1** (Context failure of a context consumer or a context provider). If a context consumer can obtain context data from the providers, we call $c$ works in acquiring context, or briefly $c$ works. Otherwise, we call the consumer $c$ fails in acquiring context, or briefly $c$ fails. If one context provider $p$ can not provide context data, we call $p$ fails in providing context, or briefly $p$ fails. Otherwise , we call $p$ works in providing context, or briefly $p$ works. We refer "work" or "fail" as two availability states of $p$.

The analysis in the previous section assumes that all context provider work, no matter how the precision of each provider. Here, we remove this assumption. In another word, it is possible that one or more context providers fail. Then expected mean of $\sigma_y^2$ given that the context consumer works is

$$E(\sigma_y^2)_{\text{works}} = \sum_{s \subseteq \hat{P}, P(s) \neq \emptyset} \left( \Pr(\text{Work}(s)) \sigma^2(s) \right) \frac{1 - \Pr(\text{Fail}(\hat{P}))}{P(P)}$$

where $\hat{P}$ is the set of context providers and each element of $\hat{P}$ can provide context information for the context consumer independently. $\text{Work}(s)$ is the predicate for the situation that all the context providers in $s \in P(\hat{P})$ work. $P(\hat{P})$ is the power set of $\hat{P}$. $\text{Fail}(\hat{P})$ is the predicate for the situation that all the context providers fail to work. $\sigma^2(s)$ is the variance of $y$ obtained by using the least square error method when the context providers in $s \in P(\hat{P})$ provide the context data simultaneously.

5.2. Context redundancy degree

Redundant context providers not only contribute to higher data precision, but also improve the reliability of the system. The acquisition middleware can provide some measurement and verification of acquisition model in an environment where the context providers are subject to failures. We propose to use redundancy degree of context providers for such a measurement and verification.

**Definition 5.2** For a context consumer $c$ and a set of context providers $\{p_1, \ldots, p_m\}$ connected with $c$, **universal redundancy degree** $r(c)$ is the maximum of $k$ such that the consumer $c$ works when any $k$ providers fail. That is

$$r(c) = \max \{ k \mid \forall k \text{ providers } p_{f_1}, \ldots, p_{f_k} (p_{f_1}, \ldots, p_{f_k} \text{ all fail } \rightarrow c \text{ works}) \}$$

**Definition 5.3** For a context consumer $c$ and a set of context providers $\{p_1, \ldots, p_m\}$ connected with $c$, **existential redundancy degree** $e(c)$ is the maximum of $k$ such that the consumer $c$ works when $k$ providers fail. That is

$$e(c) = \max \{ k \mid \exists k \text{ providers } p_{f_1}, \ldots, p_{f_k} (p_{f_1}, \ldots, p_{f_k} \text{ all fail } \rightarrow c \text{ works}) \}$$

To study some properties of the redundant context providers, we abstract a boolean expression $b(c)$ from the context acquisition model, which is called redundancy expression. For each context...
consumer \( c \), \( b(c) \) is a boolean expression on the algebraic system \(<DS(c), (\land, \lor, \neg)>\), where \( DS(c) \) is the set of context providers that the context consumer \( c \) connects with.

**Example 5.1.** If we assume any context provider \( p_i \) from a set of context providers \( \{p_1, \ldots, p_m\} \) can provide context for a context consumer \( c \) and the precision of individual provider meets the requirement, then the failure of any \( m - 1 \) context providers do not cause the failure of \( c \). In Figure 5a, the failure of only GPS location provider \( p_1 \) or only WiFi location provider \( p_2 \) will not hamper the context acquisition for the context consumer \( c \). The redundancy expression \( b(c) \) is \( p_1 \lor p_2 \).

**Property 5.1** Verifying whether a context consumer obtains data from context providers can be done in polynomial time \( O(n) \), where \( n \) is the number of context providers.

Proof. This can be done by scanning the context providers and obtaining the availability state of each context provider. If a context provider \( p_i \) fails, \( p_i = 0 \) in \( b(c) \); else \( p_i = 1 \) in \( b(c) \). Evaluate \( b(c) \). If \( b(c) = 1 \), then the context consumer \( c \) works, i.e., \( c \) can obtain data from context providers.

**Property 5.2** Finding the context providers’ availability state such that a context consumer can work is NP complete.

Proof. This is followed by the NP completeness of SAT problem.

From a practical view, negation \( \neg \) is not natural in some cases. For example, \( c \) is connected with \( p_1 \) and \( p_2 \), \( b(c) = \neg p_1 \land \neg p_2 \). This causes a paradox, i.e., with the assumption that \( c \)’s data from \( p_1 \) and \( p_2 \), the expression means when \( p_1 \) and \( p_2 \) both fail \( c \) can get data.

**Property 5.3** If a redundancy expression is constructed only with \( \land \) and \( \lor \), then the context consumer is always satisfiable.

Proof. One solution of the satisfiability is just when all the context providers work.

**Property 5.4** For one context consumer \( c \), \( e(c) \geq r(c) \).

Proof. Let \( r(c) = k_1 \). By definition 5.2, \( \forall k_1 \) providers \( p_{f_1}, \ldots, p_{f_{k_1}} \) ( \( p_{f_1}, \ldots, p_{f_{k_1}} \) all fail \( \Rightarrow c \) works), then \( \exists k_1 \) providers \( p_{f_1}, \ldots, p_{f_{k_1}} \) ( \( p_{f_1}, \ldots, p_{f_{k_1}} \) all fail \( \Rightarrow c \) works). By definition 5.3, \( e(c) \geq k_1 = r(c) \).

**Property 5.5** For one context consumer \( c \), determining \( r(c) \) according to \( c \)’s redundancy expression is \text{EXPTIME}

Proof. If there exist \( k' \) providers that fail, which causes the failure of \( c \), then \( r(c) < k' \). To determine \( r(c) \), we find the minimum of \( k' \), which can be obtained by enumerating all possible \( 2^m \) availability states of \( m \) context providers. \( r(c) \) is computed by

\[
\begin{align*}
    r(c) &= \min(k') - 1, \text{if } \min(k') > 0 \\
    r(c) &= 0, \text{if } \min(k') = 0.
\end{align*}
\]

**Property 5.6** For one context consumer \( c \), determining \( e(c) \) according to \( c \)’s redundancy expression is \text{EXPTIME}

Proof. If there exist \( k' \) providers fail but \( c \) still can obtain data, then \( e(c) \geq k' \). Hence, we can enumerate all possible \( 2^m \) states of \( m \) context providers, and then \( e(c) = \max(k') \).

However, in practice, if we restrict the redundancy expressions to disjunction normal forms (DNF) and prevent using negation, we have the following property.

**Property 5.7** If we construct the redundancy expression only with \( \land \) and \( \lor \), and translate it to DNF. The number of conjunctive clause is \( n_r \). Any context provider \( p \) is only contained in one conjunctive clause. Then
(i) \( r(c) = n_c - 1 \)

(ii) If the \( i \)th clause has \( n_i \) variables and the whole expression has \( \text{var}_{\text{num}} \) variables, \( e(c) = \text{var}_{\text{num}} - \min\{n_1, \ldots, n_n\} \).

Proof. (i) Since any context provider \( p \) is only contained in one conjunctive clause, \( n_c - 1 \) context providers are contained in at most \( n_c - 1 \) clauses. If any \( n_c - 1 \) providers fail, there is at least one clause that keeps the context consumer working. We have \( r(c) \geq n_c - 1 \). If \( r(c) > n_c - 1 \), the failure of one context provider in each clause causes the failure of the context consumer. Hence, \( r(c) = n_c - 1 \)

(ii) Without loss of generality, let \( n_j = \min\{n_1, \ldots, n_n\} \). If the providers that are not contained in \( j \)th clause fail and the providers that are contained in \( j \)th clause work, the consumer works. We have \( e(c) \geq \text{var}_{\text{num}} - \min\{n_1, \ldots, n_n\} \). Suppose \( e(c) > \text{var}_{\text{num}} - \min\{n_1, \ldots, n_n\} \), all the clauses will have failed providers, thus the consumer fails. Hence, \( e(c) = \text{var}_{\text{num}} - \min\{n_1, \ldots, n_n\} \).

E.g., for \( b(c) = (p_0 \land p_1) \lor p_2 \lor (p_3 \land p_4) \), we have \( e(c) = 4 \), \( r(c) = 2 \).

6. Provider dependency and context outliers

In this section, we discuss some more advanced issues, i.e. the dependencies among context providers and outliers of context data. Some initial methods for solving the problems are mentioned. Nevertheless, justifications of these methods deserve further investigations.

6.1. Dependencies among context providers

In a distributed network topology, the context data of a context provider may come from another context provider. In this case, to compute the coefficient of the equation \( y = \sum_{i=1}^{n} a_i x_i \), some techniques such as covariance intersection proposed by Umlmann [29] might be useful. Let \( Y \) denote the covariance matrix of \( y \), then

\[
Y = w_1 A_1^{-1} + w_2 A_2^{-1} + \ldots + w_m A_m^{-1}
\]

\[
y = Y^{-1} w_1 A_1^{-1} + \ldots + Y^{-1} w_m A_m^{-1}
\]

where \( \sum_{i=1}^{m} w_i = 1 \). Covariance intersection supports a conservative estimate of \( Y \), which is defined as \( Y \leq E\{\tilde{y}^T\} \), where \( \tilde{y} \) is the error between the measurement and the mean of \( y \).

6.2. Outliers of context data

When a provider fails to provide the data with precision it claims, the context data obtained using the above method cannot guarantee higher precision. Hence, removing the outliers in the context data is necessary sometimes. This might be done by some cluster mining methods, e.g., EM, X-Means [30]. To decide whether the data are outliers or they are in the range of claimed variance, some criteria can be used, e.g., the "covariance union" proposed by Uhlmann [29].

7. Conclusion and Outlook
In this paper, a strategy QACA is proposed to fulfill the quality assurance functionality of specification based context acquisition, wherein the system developers can pose the quality requirement on the acquisition middleware. The strategy is based on minimizing the square error, and it configures the redundant imprecise and unstable providers to obtain a more precise and stable context provision. It is shown that if there is no bound for the number of independent context providers, any high precision context requirement can be satisfied. Otherwise, there is a limitation of precision due to the cost constraints. The issues of redundancy degree, outliers, dependency among providers are also analyzed and discussed in the paper. In this paper, we use some data fusion techniques in finding quality assured configuration of context providers. In [32], a survey of data fusion techniques for reliable information was provided from a broader view. Further exploring other data fusion techniques in context aware middleware might be a future work.

8. Acknowledgements

The work is funded by National Natural Science Foundation of China (60773156), Chinese National 863 Advanced Technology Program (2008AA01Z132), Research Fund for the Doctoral Program of Chinese Higher Education (2007003089), and Chinese Major State Basic Research Development 973 Program (2011CB302200). We also thank the useful comments from Jun Wang, Weiguo Zhang and Yuji Wang.

9. References